Borrower: EYM

Lending String:
*GZM,GZM,NUI,UPM,UPM

Patron: Pettie, Seth

Journal Title: New journal of statistics and operational research.

Volume: 1 **Issue:**

Month/Year: 1969 **Pages:** 17-29

Article Author:

Article Title: Mack, C.; The Bradford method for the assignment problem

OCLC Number: 7718400

ILL # / Wiscat # - 83221717

Req 10/20

Location: SStor
Call #: (26-K-11)

Request Date: 20111020

MaxCost: $26.00IFM

Shipping Address:
University of Michigan
106 Hatcher Graduate Library - ILL
913 South University Ave.
Ann Arbor, MI 48109-1205

Fax: (734) 936-3630
Ariel: 141.211.175.21

PDF: interlibrary.loan@umich.edu

Borrowing Notes:

Copyright Compliance: CCG

ODYSSEY

Wisconsin Library Services (WiLS)

Please report all Document Delivery Problems within **48 hours** of receipt

Return this sheet via **ARIEL:** 144.92.126.152 or **EMAIL:** gzmlend@wils.wisc.edu

Briefly state the problem with the document:

alue of λ, we solve
le. If the σ_i are
that root of μ_i in
, this may not always
ly have to be consid-
lue of i, then 2^k

which satisfy 3.14

4.3

ne -ve value for λ
ues for the μ_i then
st (i.e. in the
-curve goes through
e are only one +ve
all conditions, then
i gives the limits
be improved by use
or by the numerical
.5.

Intervals for the
" J. Amer. Stat.
493
962) 'The Advanced
nd Ed) p348 eq 15.14
London)
nuity Correction in
7-337

THE BRADFORD METHOD FOR THE ASSIGNMENT PROBLEM

by C. Mack, Institute of Technology, Bradford.

Summary: This method is at once simpler and faster than any previously propounded method. It can be taught to unskilled personnel (the ability to add and subtract is needed), at the same time it needs less and much easier programming, is much quicker, and is always certain of success when used on digital computers.

Also given is a proof that such methods as have been proposed for the 2-dimensional problem will not succeed with the 3-dimensional allocation problem. Finally, methods of finding the 2nd, 3rd, etc., best solution for the 2-dimensional problem are given.

1. INTRODUCTION

The assignment (or 'allocation' or 'optimal assignment') problem is this: Given an n×n square array of numbers a_{ij}, define a 'possible' set as the set

$$a_{1,k_1},\ a_{2,k_2},\ \dots\ ,\ a_{n,k_n} \qquad 1.1$$

where $k_1, k_2, \dots k_n$ is a permutation of 1, 2, ... n. Then find that possible set which has the minimum sum (or in some cases the maximum sum). We shall call such sets 'minimal' or 'maximal'. Note that the set 1.1 consists of one element from each row but that no two of them are in the same column (hence, since there are n of them they must be in separate columns). An example of a possible set is the set of diagonal elements i.e. a_{11}, a_{22}, ... a_{nn}.

The assignment problem may arise in many practical situations. One simple example is the allocation of n jobs to n workers whose rate of production for each job is known. The optimal allocation gives the maximum rate of production. The problem has arisen in the radar systems now being developed to control the large numbers of aircraft expected to be flying simultaneously within a few years. The Royal Radar Establishment has, after careful consideration, adopted the Bradford method (see Magowan, 1965). Other organisations are adopting it (or so I believe).

Easterfield (1946 and 1961) put forward a method which the Bradford method resembles in many respects, but the latter is very much quicker. Kuhn (1955) gave a 'Hungarian' method based on existence theorems by Hungarian mathematicians. However, his method is only practicable on an electronic digital computer and requires skilful programming besides taking considerable computer time. Churchman, Ackoff, and Arnoff give a variant of the Hungarian method which is reasonably good for small arrays, but the problem of finding the least number of lines through the zeroes can be quite difficult even for a 6×6 array. (Kuhn's method, in effect, classifies these zeroes in several different classes, finally sorting them out and thus determining the minimum number of lines, but the zeroes have to be examined quite a number of times in this process.)

The Bradford method has none of these disadvantages and the technique described below can be taught to unskilled clerks in half an hour. It or its variant of Section 3 can be easily programmed (full instructions are given in Section 3), and it requires relatively little programming, takes little computer time, and is always certain of success.

The desk form of the method given in Section 2 has a simple check (most important for desk work) and with it a 15×15 array can be done in less (sometimes much less) than an hour.

All methods, directly or indirectly, are based on the observation that, if we add to or subtract from every element in a row (or column) the same amount, we do not alter the positions of the minimal or maximal set. How this is used in the Bradford method we now describe.

2. THE BRADFORD METHOD

2.1 The nature of the basic operations We shall describe the case of finding the minimal set (the finding of the maximal set is described in subsection 2.6). Now in every row there is a minimum element (for that row); though in some rows there may be alternative equal minimum elements. In the Bradford

method we raise (by
single column, or o
two or possibly mor
some of the rows.
we can select one f
in each row such th
different columns.
one such element in
form a minimal set
consists of one ele
⩾ the sum of the ro
the same positions
required answer.

This raisin
which are now descr

2.2 The operations
and an 'alternative
underline in each r
chosen arbitrarily
each underlined ele
can be seen in arra
only one base in ea
some columns and no
bases only occupy 3

(a)

Start of Step

1	12	6	25	22	20
2	18	9	18	25	27
3	13	14	21	29	28
8	8	14	21	25	24
11	16	10	22	14	25
9	21	14	12	16	15

We spread t
each 'step' by the
with more than one
(every element in)
of these bases equa
but in the same row
'alternative base'
fy this we place a

961) put forward a
esembles in many
uch quicker. Kuhn
ased on existence
ns. However, his
lectronic digital
ramming besides
Churchman, Ackoff,
ungarian method which
s, but the problem
es through the zeroes
6x6 array. (Kuhn's
e zeroes in several
them out and thus
lines, but the zeroes
of times in this

none of these dis-
ibed below can be
an hour. It or its
y programmed (full
3), and it requires
es little computer
cess.
d given in Section 2
t for desk work) and
in less (sometimes

indirectly, are based
d to or subtract from
) the same amount, we
minimal or maximal
adford method we now

OD

ations We shall
minimal set (the
cribed in subsection
a minimum element
ws there may be
s. In the Bradford

method we raise (by the same amount) the elements of a single column, or of several columns, so that we form two or possibly more alternative minimum elements in some of the rows. We keep doing this until finally we can select one from the alternative minimum elements in each row such that these chosen elements are in different columns. There must then be one and only one such element in each column. These elements must form a minimal set because any other possible set consists of one element from each row and so its sum $\geqslant$ the sum of the row minima. Finally, the elements in the same positions in the original array give the required answer.

This raising of columns is done in 'steps' which are now described.

2.2 The operations in a 'step': definition of a 'base' and an 'alternative base' If it is the first step underline in each row the minimum element (or one chosen arbitrarily if there are two or more). We call each underlined element a 'base'. This underlining can be seen in array (a). Note that there is one and only one base in each row but there may be several in some columns and none in others as in (a) where the bases only occupy 3 columns.

(a) Start of Step 1

1	12	6	25	22	20
2	18	9	18	25	27
3	13	14	21	29	28
8	8	14	21	25	24
11	16	10	22	14	25
9	21	14	12	16	15

(b) End of Step 1 = Start of 2

4	12	6	25	22	20
5	18	9	18	25	27
6	13	14	21	29	28
11	8	14	21	25	24
14	16	10	22	14	25
12	21	14	12	16	15

We spread the bases over one more column at each 'step' by the following technique: Take a column with more than one base (e.g col 1 in (a)) and raise (every element in) it by the least amount so that one of these bases equals an element in another column but in the same row. This other element is then an 'alternative base' to the original base, and to signify this we place a dot under it. Thus if we raise col 1 of (a) by 3, (when (a) becomes (b)), the original base 9 in row 6 col 1 becomes 12 and equals the 12 in row 6 col 4 (so we place a dot beneath it). Since this alternative base is in a column (col 4) containing no original bases, we stop. We then transfer the base in row 6 from its original position (col 1) to the alternative position (col 4) and thus we have spread the bases over one more column than before. The new base is underlined and the line under the original base in erased (or brackets put around it). The bases then appear as in (b) where they occupy 4 columns as against only 3 in (a). Step 1 has now ended.

The end of any previous 'step' is the start of the next 'step'. Thus we start step 2 with the array (b). Again, we raise col 1 (it has 3 bases), this time by 2, when the base 4 in row 1 becomes 6 and equals the 6 in row 1 col 3. This 6 is now an alternative base and is dotted (see (b); but we do not show the raised col 1 in (b)). This time, unfortunately, the new alternative base is in a column (col 3) already containing an original base (namely 10 in row 5). So now we raise both col 1 and col 3 by 4. the base in row 5 col 3 now goes from 10 to 14 and equals the element 14 in row 5 col 5 (it is therefore dotted). Since col 5 has no original base we stop. The array is now as in (c), and we are ready to transfer the bases.

(c)

Before Transfer at End of Step 2

10	12	10	25	22	20
11	18	13	18	25	27
12	13	18	21	29	28
17	8	18	21	25	24
20	16	14	22	14	25
18	21	18	12	16	15

(d)

Diagram showing Transfer of Bases

(x)	x	x	x	x	x
x	x	x	x	x	x
x	x	x	x	x	x
x	x	x	x	x	x
x	x	(x)	x	x	x
x	x	x	x	x	x

(_) means a removed base

→ means a new base

This transfer of bases so as to occupy one more column is shown diagrammatically in (d). Note that in col 3 a base is removed (in row 5) but a new one is formed (in row 1); while col 5 has a new base, and in col 1 one base is removed.

In general a 'step' consists of a number of raising operations. In such an operation the starting

column plus all col
formed previously d
least amount requir
We stop when the la
original base. We
occupy one more col

2.3 The Transfer R
though it is very ea
(d) above). It is

The last formed alt
the original base i
an alternative " "
the original " "
an alternative " "

and so on; until a
contains other (ori
This succee
in a column which h
from a column which
intermediately have
a new base in anoth
Note that n
will necessarily ap
(g) below); (ii) sc
alternative bases b
above process (this

2.4 The Short Form
to add to elements
(because addition c
So the following Sh
confines addition t
out all operations
The array (
going from (b) to (
ations clear. In
that are raised in
by raising col 1.
the bases of col 1
in A is the amount
bases then become 6
the element 6 in rc

s (b)), the original
d equals the 12 in
ath it). Since
n (col 4) containing
en transfer the base
(col 1) to the
us we have spread
before. The new
er the original base
t). The bases then
4 columns as against
ied.
tep' is the start of
tep 2 with the array
as 3 bases), this time
omes 6 and equals the
an alternative base
not show the raised
unately, the new
ol 3) already contain-
row 5). So now we
the base in row 5
quals the element 14
tted). Since col 5
e array is now as in
the bases.

ing
Bases

x x
x x (_) means a
x x removed base
x x
x x → means a
x x new base

as to occupy one more
in (d). Note that in
) but a new one is
s a new base, and in

sts of a number of
peration the starting

column plus all columns containing alternative bases formed previously during the step are raised by the least amount required to form a new alternative base. We stop when the latter is in a column containing no original base. We then transfer the bases so that they occupy one more column by the following rule.

2.3 The Transfer Rule We now give this in words, though it is very easily demonstrated graphically (see (d) above). It is

The last formed alternative base is always a new base
the original base in the above element's row is removed
an alternative " " " " " col is a new base
the original " " " " " row is removed
an alternative " " " " " col is a new base

and so on; until a base is removed from a column which contains other (original) bases.

This succeeds because (i) a new base is formed in a column which had none before; (ii) a base is removed from a column which had more than one; (iii) columns intermediately have a base removed from one row but have a new base in another row.

Note that not all columns carrying alternatives will necessarily appear in the above transfer (see array (g) below); (ii) some columns may carry two or more alternative bases but only one is selected during the above process (this can be done arbitrarily, see (k)).

2.4 The Short Form Now there is no need, during a step, to add to elements other than the bases in a column (because addition cannot make a non-minimum a minimum). So the following Short Form has been devised which (i) confines addition to essential numbers only, (ii) sets out all operations simply and clearly.

The array (e) below shows the above Step 2 (i.e. going from (b) to (c)) in Short Form and makes its operations clear. In (e) the line C gives the columns that are raised in chronological order. Thus we start by raising col 1. Below the 1 (in line C) we place the bases of col 1 (namely 4,5,6). The first amount in A is the amount by which col 1 is raised. The bases then become 6,7,8 as shown. But the 6 equals the element 6 in row 1 col 3, so a dot is placed under

(e)

Step 2 in Short Form

						C:	1	3	5
						A:	2	4	
4	12	6	25	22	20		4	6	10
5	18	9	18	25	27		5	7	11
6	13	14	21	29	28		6	8	12
11	8	14	21	25	24				
14	16	10	22	14	25			10	14
12	21	14	12	16	15				
6	4					:T			

(f)

Transfer of Bases

(4) →	6	10
5	7	11
6	8	12
	(10) →	14

both these elements. Since the alternative base is in col 3 then 3 is inserted in line C and the base in col 3 (namely 10 in row 5) is added in the appropriate place below the 3 (in line C). The second amount 4 in line A is that by which both col 1 and col3 are raised. Their bases now become 10,11,12 and 14 (see the last column on the right in (e)). The last element equals the 14 in row 5 col 5 and a dot is placed under both. Since col 5 has no (original) base we stop.

The transfer of bases can then be marked in very simply (it is shown in (f) above, to avoid too much marking of (e), but it would normally be marked on (e)).

After this marking of the transfer, the amount T nominally added to each element of a column during the complete step is totalled and actually added to that column. The rule is this: T = Total amount shown in line A to the right of where the column is mentioned in line C; (thus, for col 3, T = 4; for col 1, T = 4+2 = 6). The bases are then marked in using the markings on the right, and the array is ready to start the next step, (the array resulting from (e) is shown in (g)).

2.5 Further Shortening of the Process If the working is carried out in pencil then the augmented columns which arise from the addition of the T can be written (with the correct bases underlined) next to their original columns and the latter erased. When this has been done the working on the right is erased. This saves having to write out complete arrays (a disadvantage of the Hungarian methods).

Further si
are: (i) prime (or
columns being rais
not being raised (
'smallest elements
are being raised;
would be confusing
(ii) the minim
element' and the c
on the right gives
elements 20,18,21,
under the 3 in lin
A value is min{20-
(iii) draw a ve
number appears in
seen at a glance)
(iv) where and
through a primed e
in another column
By the full
20x20 arrays can b

(g)

Step 3 in Short Fo

10	12	10	25	22	20'
11	18	13	18'	25	27
12	13	18	21'	29	28
17	8	18	21'	25	24
20	16	14	22	14	25
18	21	18	12	16	15
10	9	8	3		:

The array (
for our illustrati
bases shown on the
to their columns w
there is no need t
we are only intere
(and (j) shows the
However, the final
in that they enabl

(f)
ransfer of Bases

)→6 10
7 11
8 12

(10)→14

tive base is in
he base in col 3
ropriate place
ount 4 in line A
raised. Their
e last column on
uals the 14 in
th. Since col

e marked in very
id too much
marked on (e)).
fer, the amount T
lumn during the
added to that
amount shown
umn is mentioned
col 1, T = 4+2 =
ng the markings
start the next
shown in (g)).

If the working
nted columns which
written (with
heir original
this has been done
This saves having
antage of the

Further simple but valuable time-saving devices are: (i) prime (or tick), in each row of the bases of the columns being raised, the smallest element in a column not being raised (in (g) below, we show the primed 'smallest elements' at the point where cols 1,2,and 3 are being raised; the full priming throughout the step would be confusing to show)

(ii) the minimum difference between primed 'smallest element' and the corresponding (augmented) base value on the right gives the A value (thus, with the primed elements 20,18,21,21 shown in (g) the augmented bases under the 3 in line C are 10,13,14,9, and so the next A value is $\min\{20-10, 18-13, 21-14, 21-9\} = 5$)

(iii) draw a vertical line through any column whose number appears in line C (columns being raised can then be seen at a glance)

(iv) where and only where this vertical line passes through a primed element then another 'smallest element' in another column must be found .

By the full short technique any 15×15 and some 20×20 arrays can be solved within an hour.

(g)

Step 3 in Short Form

A: 1 1 5 3

						C:1	2	3	4	6
10	12	10	25	22	20'			10	15	18
11	18	13	18'	25	27	(11)→	~~12~~ →	13 →	18	21
12	13	18	21'	29	28	12	13	14	19	22
17	8	18	21'	25	24		8	9	14	17
20	16	14	22	14	25					
18	21	18	12	16	15				(12)→	15
10	9	8	3		:T					

The array (g) shows in Short Form, the last step, for our illustrative array with the last transfer of bases shown on the right. When the T values are added to their columns we obtain the array (h). Actually, there is no need to add the final step T values because we are only interested in the positions of the bases (and (j) shows the minimal set for the original array). However, the final T-values are themselves of interest in that they enable a simple check to be carried out.

(f)
Transfer of Bases

```
(4)→6    10
 5   7   11
 6   8   12

  (10)→14
```

alternative base is in
C and the base in col 3
the appropriate place
econd amount 4 in line A
col3 are raised. Their
(see the last column on
ement equals the 14 in
under both. Since col

n then be marked in very
, to avoid too much
mally be marked on (e)).
e transfer, the amount T
of a column during the
tually added to that
Total amount shown
the column is mentioned
= 4; for col 1, T = 4+2 =
i in using the markings
ready to start the next
(e) is shown in (g)).

Process If the working
he augmented columns which
T can be written (with
ext to their original
When this has been done
sed. This saves having
disadvantage of the

Further simple but valuable time-saving devices are: (i) prime (or tick), in each row of the bases of the columns being raised, the smallest element in a column not being raised (in (g) below, we show the primed 'smallest elements' at the point where cols 1,2,and 3 are being raised; the full priming throughout the step would be confusing to show)

(ii) the minimum difference between primed 'smallest element' and the corresponding (augmented) base value on the right gives the A value (thus, with the primed elements 20,18,21,21 shown in (g) the augmented bases under the 3 in line C are 10,13,14,9, and so the next A value is min{20-10,18-13,21-14,21-9} = 5)

(iii) draw a vertical line through any column whose number appears in line C (columns being raised can then b[e] seen at a glance)

(iv) where and only where this vertical line passes through a primed element then another 'smallest element' in another column must be found .

By the full short technique any 15×15 and some 20×20 arrays can be solved within an hour.

(g)
Step 3 in Short Form

```
                          C:1    2    3    4    6
                          A:  1    1    5    3
10 12 10 25 22 20'     |              10   15   18
11 18 13 18'25 27      | (11)—12 —13→18        21
12 13 18 21'29 28      |  12     13   14   19   22
17  8 18 21'25 24      |          8    9   14   17
20 16 14 22 14 25      |
18 21 18 12 16 15      |           (12)→15
10  9  8  3        :T
```

The array (g) shows in Short Form, the last step, for our illustrative array with the last transfer of bases shown on the right. When the T values are added to their columns we obtain the array (h). Actually, there is no need to add the final step T values because we are only interested in the positions of the bases (and (j) shows the minimal set for the original array). However, the final T-values are themselves of interest in that they enable a simple check to be carried out.

(h)
Final (and Check) Array

	20	21	18	28	22	20
	21	27	21	21	25	27
	22	22	26	24	29	28
	27	17	26	24	25	24
	30	25	22	25	14	25
	28	30	26	15	16	15
W:	19	9	12	3	0	0

(j)
Original Array showing Minimal Set

1	12	6	25	22	20
2	18	9	18	25	27
3	13	14	21	29	28
8	8	14	21	25	24
11	16	10	22	14	25
9	21	14	12	16	15

It is to be noted that in the transfer of bases shown in (g), only 2 out of the 4 alternative bases (shown dotted on the right) appear as new bases. Note also that in the minimal set shown in (j) the two smallest elements 1 and 2 of the array do not appear.

2.6 The Check The line W in (h) gives the sum of the T values added to the columns in the whole process. If these W values are added to the original columns then we should get the final array. If the row minima in this final array are in the positions given by the final step we have made no mistake (not in finding the minimal set; there might be other mistakes but these do not matter).

2.7 The Maximal Set This can be found in exactly the same way as the minimal set except that we underline the row maximum as a base; and in a step we lower those columns which have more than one base to form alternative bases.

In (k) below we show a single step in Short Form.

(k)
A Short Form Step in Finding a Maximal Set

	24	13	11	20	10	14
	26	11	24	15	8	16
	11	12	28	14	7	13
	9	10	9	16	5	18
	8	9	21	11	9	21
	7	18	13	19	4	10
T:	-5		-3	-1		-3

C: 1 3 6 4
A: -2 0 -2 -1

24	22	22	20	19
26	24	24	22	21
	28	28	26	25
		(18) →	16	15
	(21) →	21	19	18
			(19) → 18	

There are some fea
col 3 is brought i
must be taken to e
3 in line C on the
column on the righ
simultaneously (1,
one and one only o
arbitrarily, since
which will shorten

3. ALTERNATIVE FOR

3.1 Basic Operatio
grammed without di
columns being rais
tronic computer, w
is perhaps more ea
In this fo
each step by selec
we lower it by the
elements to equal
alternative base.
a column containin
this and the first
base; and so on.
step on the array
in the previous me
occupy one more co

3.2 The Transfer R
The (original) bas
an alternative bas
the original "
an alternative "
and so on
until a new base is
at the start of the
This succe
umn containing the
bases removed, int
base, and the firs

(j)
inal Array showing
Minimal Set

6 25 22 20
9 18 25 27
14 21 29 28
14 21 25 24
10 22 14 25
14 12 16 15

ansfer of bases shown in
tive bases (shown dotted
s. Note also that in
two smallest elements
ar.

gives the sum of the
the whole process.
he original columns
ay. If the row minima
positions given by the
ke (not in finding the
r mistakes but these do

found in exactly the
ept that we underline
in a step we lower those
e base to form alternative

single step in Short Form.

3 6 4
0 -2 -1
2 22 20 19
4 24 22 21
8 28 26 25
(18)→16 15
1)→21 19 18
(19)→18

There are some features worth noting in (k). (i) When col 3 is brought in, it has two original bases and care must be taken to ensure that both are recorded below the 3 in line C on the right. (ii) In the last but one column on the right two alternative bases have been forme simultaneously (16 and 20); during the transfer of bases one and one only of these has to be selected. We do thi arbitrarily, since there is no means of telling beforehan which will shorten the subsequent work.

3. ALTERNATIVE FORM OF BRADFORD METHOD: PROGRAMMING

3.1 Basic Operations The previous method can be programmed without difficulty; but, since addition to the columns being raised presents no difficulty on an electronic computer, we now give an alternative form, which is perhaps more easily programmed.

In this form, to find the minimal set, we start each step by selecting a column containing no bases, and we lower it by the least amount for one of the column's elements to equal a base. This element becomes an alternative base. If the (original) base equalled is in a column containing other bases we stop; if not we lower this and the first column to form another alternative base; and so on. (The result of carrying out such a step on the array (b) is shown in (m) below.) Just as in the previous method we have to transfer the bases to occupy one more column which we do thus:

3.2 The Transfer Rule
The (original) base last equalled is removed
an alternative base in the above's row is a new base
the original " " " " col " removed
an alternative " " " " row " a new base
and so on
until a new base is formed in the column first lowered at the start of the step.

This succeeds, just as in 2.3, because the column containing the last equalled base has one of its bases removed, intermediate columns lose one and gain one base, and the first column lowered gains one.

3.3 The Programme

We need the following quantities during the programme (which is for the minimal set) :

- a_{ij} current value of the element in the ith row and jth column of the array
- n_j the <u>number of</u> bases in the jth column
- b_i the value of the <u>base</u> in the ith row, and
- c_i the <u>number</u> (i.e. the j-value) of its column
- a_i the <u>number</u> of the column containing the <u>alternative</u> base in the ith row (if none, put $a_i = 0$)
- $w_1, w_2, \ldots w_k$ the number<u>s</u> of the columns being lo<u>w</u>ered during the kth lowering operation of a step

We commence by finding b_i and c_i for the original array, and from the c_i we find the n_j. We are then ready for step 1. A step is <u>started</u> thus:

1) put all $a_i=0$, and put $w_1=J=$ first j whose $n_j= 0$

Then we begin the <u>lowering</u> operations. At the kth such operation

2) find $M = \min(a_{ij} - b_i)$ for all i whose $a_i=0$ and for $j = w_1, \ldots w_k$; suppose $i=i'$, $j=j'$ at this minimum
3) put $a_{i'}= j'$; subtract M from <u>all</u> a_{ij} for $j=w_1,\ldots w_k$
4) if $n_{c_{i'}} = 1$, put $w_{k+1}= c_{i'}$, $k = k+1$, jump to 2)
5) if $n_{c_{i'}} > 1$, subtract 1 from $n_{c_{i'}}$, put $n_J = 1$,

Then effect the <u>transfer</u> of bases thus: put $i_1=$last i', at the tth transfer

6) put $c_{i_t} = a_{i_t}$; if $a_{i_t} = J$ jump to 7), if not find i_{t+1} such that $c_{i_{t+1}} = a_{i_t}$; put $t=t+1$, jump to 6)
7) set all $b_i = a_{ij''}$ where $j'' =$ (new) a_i ; jump to 1)

When no $n_j = 0$ (when all n_j should be 1), the c_i = column number of the element of the minimal set in the ith row.

The programme for the maximal set is much the same, though then the b_i are the maxima in their rows, $M = \min (b_i - a_{ij})$, and M is added not subtracted in 3).

We now show some of the details of the working in carrying out a step by the above programme, starting

with the array of

A Step of the Alte
Values of t

Starting Value

i or j :	1	2	3
n_j	3	1	1
b_i	4	5	6
c_i	1	1	1
1) a_i	0	0	0
		J = 5	

Before Transfe

n_j	2	1	1
a_i	3	0	0

After Transfer

c_i	3	1	1
b_i	4	5	6

(

Final Re

4	12	<u>4</u>
<u>5</u>	18	7
<u>6</u>	13	12
11	<u>8</u>	12
14	16	8
12	21	12

4. THE 3-DIMENSION

We shall sh
of raising or lower

wing quantities
the minimal set) :
the ith row and

column
th row, and
of its column
ining the alternative
put $a_i = 0$)
olumns being lowered
ion of a step
r the original array,
e are then ready for

st j whose $n_j = 0$
ns. At the kth

whose $a_i = 0$ and for
at this minimum
ll a_{ij} for $j = w_1, \dots w_k$
=k+1 , jump to 2)
, put $n_J = 1$,
i'
thus: put i_1=last i',

to 7), if not

; put t=t+1, jump to 6)

ew) a_i ; jump to 1)
d be 1), the c_i =
e minimal set in the

mal set is much the
maxima in their rows,
not subtracted in

etails of the working
ve programme, starting

with the array of (b) given earlier.

(ℓ)

A Step of the Alternative Method (applied to (b))

Values of the Working Variables

Starting Values

i or j :	1	2	3	4	5	6
n_j	3	1	1	1	0	0
b_i	4	5	6	8	10	12
c_i	1	1	1	2	3	4
1) a_i	0	0	0	0	0	0
		J = 5				
Before Transfer						
n_j	2	1	1	1	1	0
a_i	3	0	0	0	5	5
After Transfer						
c_i	3	1	1	2	5	4
b_i	4	5	6	8	8	10

During Lowering

	k :	1	2	3
	w_k	5	3	4
2)	M	4	0	2
	i'	5	6	1
	j'	5	5	3
3)	$a_{i'}$	5	5	3
4)	$c_{i'}$	3	4	1
	$n_{c_{i'}}$	1	1	3 > 1
Transfer				
	t :	1	2	
	i_t	1	5	
	c_{i_t}	3	5 = J	

(m)

Final Result of the above Step

4	12	4	23	16	20
5	18	7	16	19	27
6	13	12	19	23	28
11	8	12	19	19	24
14	16	8	20	8	25
12	21	12	10	10	15

4. THE 3-DIMENSIONAL ALLOCATION PROBLEM

We shall show by an example that the technique of raising or lowering columns, rows, etc., will not succeed, in general, here.

The general element in an array here is a_{ijk}. The 2-dimensional array formed when i =constant we shall call a row; when j = constant, a column; and when k = constant a 'level'. Consider the following 2×2×2 array:

	Level 1		Level 2	
Row 1	0	-4	1	1
Row 2	1	1	10	0
	Col 1	Col 2	Col 1	Col 2

The minimal set is 0, 0 as trial and error will show. We shall show that by adding r_1, r_2 to the rows and c_1, c_2 to the columns we cannot make the elements in the above ρ,C, positions simultaneously zero. The modified array is:

$$\begin{array}{cccc} \underline{r_1+c_1} & -4+r_1+c_2 & r_1+c_1+1 & r_1+c_2+1 \\ r_2+c_1+1 & r_2+c_2+1 & r_2+c_1+10 & \underline{r_2+c_2} \end{array}$$

If the underlined elements are minima in their levels, then it follows that

$$c_2 - c_1 - 4 > 0 \quad (1) \qquad r_2 - r_1 + c_2 - c_1 + 1 > 0 \quad (2) \qquad r_2 - r_1 + 1 > 0 \quad (3)$$
$$c_1 - c_2 + 10 > 0 \quad (4) \qquad r_1 - r_2 + c_1 - c_2 + 1 > 0 \quad (5) \qquad r_1 - r_2 + 1 > 0 \quad (6)$$

From (1) and (5), $r_1 - r_2 > c_2 - c_1 - 1 > 4 - 1 = 3$
From (3) $1 > r_1 - r_2$
and the results are incompatible.

Similarly, adding to the rows and levels will not make the underlined elements minima in their columns, and adding to the levels and columns will not be successful either.

An intuitive way of seeing that such methods are unlikely to be successful is this: In the 2-dimensional array we have n^2 elements and (in effect) 2n quantities we can add, n to the columns and n to the rows. In the 3-dimensional case we have n^3 elements and only 3n quantities we can add. This is an order of magnitude too small.

5. THE 2ND, 3RD, ETC. BEST 2-DIMENSIONAL SOLUTION

However, it is possible to provide answers to these problems. We can find the 2nd best minimal solution this way: Take the minimal set, and replace each element of this set in turn by a large number

(say the sum of t
set for each of t
set is the 2nd be
long as might be t
only slightly fro
possible for the s
element with the l

0
100
1 1

the best solution

The 3rd be
minimal set for th
common element of
(ii) making a non-
common element of
The smallest of th
in an exceptional
found. The situa
can be found along

RE

CHURCHMAN,C.W., .
duction to

EASTERFIELD,T.E.
J. Lond
& (1961) 'An
Operat. Res
KUHN,H.W., (1955
ment Proble
MAGOWAN,S., (196
tion' Monog

ray here is a_{ijk}.
i =constant we shall
umn; and when k =
ollowing 2×2×2 array:

2
1
0
ol 2

error will show.
o the rows and c_1, c_2
ements in the above
The modified array

r_1+c_2+1
) r_2+c_2
ma in their levels,

+1>0 (2) $r_2-r_1+1 > 0$ (3)
+1>0 (5) $r_1-r_2+1 > 0$ (6)
> 4 - 1 = 3

ows and levels will
nima in their columns,
s will not be success

that such methods
is: In the 2-dim-
and (in effect) 2n
umns and n to the rows.
^{3} elements and only
an order of magnitude

ONAL SOLUTION

provide answers to
2nd best minimal
l set, and replace
a large number

(say the sum of the row maxima). We find the minimal set for each of these modified arrays and their minimal set is the 2nd best solution. This will not take so long as might be thought, for the arrays will differ only slightly from the original array. However, it is possible for the second-best solution to have no common element with the best solution. In this array

0	1	100
100	0	1
1	100	0

the best solution is 0,0,0 and the 2nd best 1,1,1.

The 3rd best solution is found by finding the minimal set for the arrays formed by (i) making any common element of 1st and 2nd best solutions very large, (ii) making a non-common elemnt of the 1st and a non-common element of the 2nd simultaneously very large. The smallest of these solutions is the 3rd best. But, in an exceptional case ${}^nC_2$ minimal sets might have to be found. The situation is worse for the 4th best which can be found along similar lines.

REFERENCES

CHURCHMAN, C.W., ACKOFF, R.L., and ARNOFF, E.L., "Introduction to Operations Research" Chap 12, John Wiley, New York.

EASTERFIELD, T.E., (1946) 'A Combinatorial Algorithm' J. London Math. Soc., 21, 219-226
& (1961) 'An Algorithm for the Allocation Problem' Operat. Res. Quart., 11, No 3 (Sept)

KUHN, H.W., (1955) 'The Hungarian Method for the Assignment Problem' Nav. Res. Log. Quart., 2, 83-98

MAGOWAN, S., (1965) 'A Method of Plot to Track Correlation' Monograph 2153, Royal Radar Establishment, Malvern, England.